REMARKS

Claims 1-56 were presented for examination. Claims 1-56 stand rejected. Applicants submit that claims 1-56 are in condition for allowance. The following comments address all stated grounds of rejection. Applicants respectfully urge the Examiner to pass the claims to allowance in view of the remarks set forth below.

Claim Rejections Under 35 USC §103

I. Claims 1-20 and 29-48 Stand Rejected Under 35 U.S.C §103(a)

Claims 1-20 and 29-48 stand rejected under 35 U.S.C. §103 as unpatentable over Shannon et al., "Mapping the Interface Description Language Type Model in C," November 1989, IEEE Transactions on Software Engineering, Vol. 15, No. 11 ("Shannon"), in view of Research Systems, "IDL", copyright 1994 ("Research Systems").

Claims 1 and 29

Shannon and Research Systems, alone or in combination, fail to teach or suggest each and every element of Independent claims 1 and 29. Claims 1 and 29 are directed towards a method and computer program product, respectively. These claims recite processing a definition of a function associated with a first language to create description information about the function. The description information being sufficient to enable *translation of a call to the function into a call to a corresponding function in a second source code language* without requiring processing of the definition of the function.

Shannon discusses mapping the definition of a data structure in the IDL into a corresponding data structure in a target language (specifically C). Applicant respectfully disagrees with the Examiners assertion that data structures read on a function in a first language because data structure is inherent to function of language. Functions and data structures are not the same thing. Data structures, in the case of IDL refer to four basic types (Boolean, Integer, Rational, and String) and four structure types (nodes, classes, sets, and sequences). Functions, on the other hand, act on structures (For example, adding two integers together). As such, just because a data structure is mapped from one language to another (i.e. an integer in IDL to an Integer in C) does not mean the functions in one language are mapped to another. Indeed, the mapping of functions that act upon data structures in one language is not mentioned, discussed, or contemplated by Shannon. Shannon only concerned that data structures of IDL can be mapped to C so that the functions of C will work on the mapped data structures.

Applicants further believes the Examiner is confused as to what is being claimed. Namely, the Examiner seems to be confusing the *translation of a function* with *translation of a call to a function to a call to a corresponding fucntion.* These are not the same thing. The present invention does not require the definition of function in the first language to be translated into a definition of the function the second language. Even if one where to accept the Examiners construction that data structures read on functions, the translation of a function from one language to another language is not what is being claimed. Claims 1 and 29 claim *translation of a call* not the translation of a function. That is, the second language may already have a function that corresponds to the function in the first language in which case, to achieve the same result, it is only necessary to replace the call to the function in the first language with the call to the corresponding function in the second language.

The Examiner has admitted that Shannon fails to disclose processing the definition of a function associated with the first language to create the description information.

In the Final Office Action, the Examiner indicates that Shannon does not disclose processing a definition of a function associated with the first source code language to create the description information. The Examiner cites Research Systems for the purpose of suggesting that one ordinarily skilled in the art might modify the IDL of Shannon to incorporate the IDL of Research Systems to process a definition of a function provided by the first source code language to create the description information. However, Research Systems fails to address the deficiencies of the Shannon reference.

Applicants believe the Examiner may be confused the identical acronyms (IDL) used for two totally different products. In Shannon, IDL stands for Interface Description Language which is a notation for describing the characteristics of data structures. In Research Systems, IDL stands for Interactive Data Language which is a programming language marketed by Research Systems. Although, the use of the same acronym (IDL) can understandably cause confusion (see attached wikopedia entry) the Interface Description Language of Shannon is not related to the Interactive Data Language of Research Systems. The Interactive Data Language of Research Systems is a programming language used by scientist for processing data. As such it is not surprising that is provides math and image functions. But this programming language is not an Interface Description Language as discussed in Shannon used to describe data structures. As such there is no motivation to combine Shannon and Research Systems because they discuss two entirely different and unrelated products and methodologies.

Furthermore, Research Systems does not teach or suggest description information being sufficient to enable translation of a call to the function into a call to a corresponding function in a

second language without requiring processing of the definition of the function. The description of the programming language of IDL of Research Systems does not discuss the function call translation of the claimed invention between a function in one language that corresponds to a plurality of functions in another language. Thus, Research Systems fails to bridge the deficiencies of the Shannon reference.

Claims 2-10 and 30-38

For at least the above-discussed reasons, Shannon in view of Research Systems fails to detract from the patentability of claims 1 and 29. Claims 2-10 depend on and incorporate the patentable subject matter of independent claim 1. Claims 30-38 depend on and incorporate the patentable subject matter of independent claim 29. As such, Shannon in view of Research Systems also fails to detract from the patentability of claims 2-10 and 30-38. Accordingly, Applicants respectfully request the Examiner to reconsider and withdraw the rejection of claims 1-10 and 29-38 under U.S.C §103.

Claims 11 and 39

Independent claims 11 and 49 are directed towards a method and computer program product respectively. These independent claims recite providing a file of description items and using the file of description items to translate a first program file into a second program file. Each item of the description items includes description information about a definition of a function defined by a first source code language. The description information being sufficient to enable translation of a call to the function in the first source code language into a call to a corresponding function in a second source code language without requiring processing of the

definition of the function. The function in the first source code language corresponds to a plurality of functions in the second source code language. Applicants submit that Shannon in view of Research Systems does not teach or suggest each and every feature of the claimed invention.

For the reasons set forth above in regard to claims 1 and 29, Shannon does not teach or suggest description information being sufficient to enable translation of a call to the function in a first language into a call to a corresponding function in a second language without requiring processing of the definition of the function. As stated above, Shannon does not discuss the translation of a call at all.

As stated above in regard to claims 1 and 29, Research Systems fails to address the deficiencies of the Shannon reference. As with Shannon, Research Systems does not teach or suggest description information being sufficient to enable translation of a call to the function in a first language into a call to a corresponding function in a second language without requiring processing of the definition of the function. Thus, Research Systems fails to bridge the deficiencies of the Shannon reference. Furthermore, there is no motivation to combine Shannon and Research Systems because they are directed to independent unrelated products.

Claims 12-20 and 40-48

For at least the above-discussed reasons, Shannon in view of Research Systems fails to detract from the patentability of claims 11 and 39. Claims 12-20 depend on and incorporate the patentable subject matter of independent claim 11. Claims 40-48 depend on and incorporate the patentable subject matter of independent claim 39. As such, Shannon in view of Research Systems also fails to detract from the patentability of claims 12-20 and 40-48. Accordingly,

Applicants respectfully request the Examiner to reconsider and withdraw the rejection of claims 11-20 and 39-48 under U.S.C §103.

Claims 21-28 and 49-56 Stand Rejected Under 35 U.S.C §103(a)

Claims 21-28 and 49-56 stand rejected under 35 U.S.C. §103 as unpatentable over Elmroth et al., "A Web Computing Environment for the SLICOT Library," December 2000, Brite-Euram III, Networks Programme NICONET ("Elmroth") in view of Research Systems and in further view of Shannon.

Claims 21 and 49

Independent claims 21 and 49 are directed towards a method and computer program product respectively. These independent claims recite providing a library file including functions defined by a first source code language, and processing the library file to create a function library and a description file. The function library includes one or more functions defined by a second source code language and each function in the function library being a translated version of a function in the library file. The description file includes description information about each function in the library file. The description information being sufficient to enable translation of a call to the function in a first source code language into a call to a corresponding function in the second source code language without requiring processing of the definition of the function. These independent claims further recite using the description file to translate a program file from the first source code language into the second source code language, wherein each call in the program file to a function in the library file is translated into a call to a corresponding function in the second source code language. Applicants submit that Elmroth in

view of Research Systems and in further view of Shannon does not teach or suggest each and every feature of the claimed invention.

Neither Elmroth, Research Systems, nor Shannon, alone or in combination teach or suggest description information being sufficient to enable translation of a call to the function into a call to a corresponding function in a second source code language without requiring processing of the definition of the function. In the Final Office Action, the Examiner indicates Elmroth does not teach or suggest description information being sufficient to enable translation of a call to the function into a call to a corresponding function in a second source code language without requiring processing of the definition of the function. The Examiner cites Shannon and Research Systems to address the deficiencies of the Elmroth reference. However, for the reasons discussed above in connection with the rejection of independent claims 1, 11, 21, and 39, Shannon in view of Research Systems fail to teach or suggest description information being sufficient to enable translation of a call to the function in a first source code language into a call to a corresponding function in a second source code language without requiring processing of the definition of the function, and wherein the function in the first source code language corresponds to a plurality of functions in the second source code language. Therefore, Elmroth in view of Research Systems and in further view of Shannon fail to teach or suggest each and every feature of the claimed invention.

Claims 22-28 and 50-56

For at least the above-discussed reasons, Elmroth in view of Research Systems in further view of Shannon fails to detract from the patentability of claims 21 and 49. Claims 22-28 depend on and incorporate the patentable subject matter of independent claim 21. Claims 50-56

depend on and incorporate the patentable subject matter of independent claim 49. As such, Elmroth in view of Research Systems in further view of Shannon also fails to detract from the patentability of claims 22-28 and 50-56. Accordingly, Applicants respectfully request the Examiner to reconsider and withdraw the rejection of claims 21-28 and 49-56 under U.S.C §103.

CONCLUSION

In view of the amendments and remarks set forth above, Applicants contend that the presently pending claims in this application are patentable and in condition for allowance. Therefore, Applicants respectfully urge the Examiner to pass the claims to allowance.

If the Examiner deems there are any remaining issues, we invite the Examiner to call the Applicants' Attorney at the telephone number identified below.

Respectfully submitted,

LAHIVE & COCKFIELD, LLP

Dated: **November 22, 2005**

James McKenzie
Registration No. 51,146
Attorney for Applicants

Lahive & Cockfield, LLP
28 State Street
Boston, Massachusetts 02109
(617) 227-7400 (Tel.)
(617) 742-4214 (Fax)

IDL programming language

From Wikipedia, the free encyclopedia.

IDL, short for **interactive data language**, is a programming language which is a popular data analysis language among scientists. IDL is often confused, at the acronym level, with another (unrelated) "IDL": the Interface description language.

Contents

Overview

IDL is vectorized, numerical, interactive, Fortran-like, and is commonly used for interactive processing of large amounts of data (including image processing), right from the keyboard.

IDL originated from early VAX/VMS/Fortran, and its syntax still shows its heritage:

```
x = findgen(100)/10
y = sin(x)/x
plot,x,y
```

Note that the operation in the second line applies in a vectorized manner to the whole 100-element array created in the first line, analogous to the way general-purpose array programming languages (such as APL or J) would do it.

As most other array programming languages, IDL is very fast doing vector operations (sometimes as fast as a well-coded custom loop in FORTRAN or C) but quite slow if elements need processing one-by-one. Hence part of the art of using IDL (or any other array programming language, for that matter) for numerically heavy computations is to make use of the inbuilt vector operations.

History

The predecessor versions of IDL were developed in the 1970s at the Laboratory for Atmospheric and Space Physics (LASP) at the University of Colorado at Boulder. At LASP David Stern was was involved in efforts to allow scientists to test hypotheses without employing programmers to write or modify individual applications. The first program in the evolutionary chain to IDL that Stern developed was named Rufus, a simple vector oriented calculator that ran on the PDP-12. It accepted two-letter codes that specified an arithmetic operation, the input registers to serve as operands, and the destination register. A version of Rufus developed on the PDP-8 was the Mars Mariner Spectrum Editor (MMED). MMED was used by LASP scientists to interpret data from Mariner 7 and Mariner 9. Later, Stern wrote a program named SOL, which also ran on the PDP-8. Unlike its predecessors,

it was a true programming language with a FORTRAN-like syntax. SOL was an array-oriented with some primitive graphics capabilities.

Stern left LASP to found Research Systems Inc. (RSI) in 1977. The first RSI product was IDL for the PDP-11. In this release, the graphics supported by IDL were primarily Tektronix terminals and raster graphics displays. RSI sold its first IDL licenses to NASA's Goddard Space Flight Center and Ball Aerospace in 1979. Two years later RSI released an initial VAX/VMS version of IDL, which was written in VAX-11 MACRO and FORTRAN. It took advantage of the VAX virtual memory and 32-bit address space. The National Center for Atmospheric Research (NCAR), the University of Michigan, the University of Colorado, and the Naval Research Laboratory started to use IDL with this version.

In 1987 RSI shifted development work of IDL to the Unix environment, which required a complete re-write of the code in C rather than a port of the existing version of VAX IDL. Stern and Ali Bahrami rewrote IDL for Unix on the Sun 3, taking advantage of the re-write to extend and improve the language. Subsequently, IDL was further expanded and ported to several variants of Unix, VMS, Linux, Microsoft Windows (1992), and MacOS (1994).

Widgets were added to IDL in 1992, providing event-driven programming with graphical user interfaces. In 1997 ION (IDL On the Net), a web server-based system, was commercially released. The first version of ENVI, an application for remote sensing multispectral and hyperspectral image analysis written in IDL, was released in 1994.

IDL has been applied widely in space science. The European Space Agency used IDL to process almost all of the pictures of Halley's Comet taken by the Giotto spacecraft. The team repairing the Hubble Space Telescope used IDL to help them diagnose anomalies in the lens. In 1995 astronauts on board a space shuttle used IDL loaded on a laptop to study ultraviolet radiation.

Features

As a computer language, IDL:

- is dynamically typed
- has a single namespace
- was originally single threaded but now has many multi-threaded functions and procedures
- has all function arguments passed by reference ("IN-OUT")
- does not require variables to be predeclared
- provides only COMMON block declarations to share global values among routines
- provides a basic form of object-oriented programming, somewhat similar to Smalltalk
- compiles to an interpreted, stack-based intermediate p-code (à la Java VM)
- provides a simple and efficient index slice syntax to extract data from large arrays
- provides various integer sizes, as well as single and double precision floating point real and complex numbers
- provides composite data types such as character strings, homogeneous-type arrays, and simple (non-hierarchical) record structures of mixed data types

Problems

Some of these features, which make IDL very simple to use interactively, also cause difficulties when building large programs. The single namespace is particularly problematic in those cases. IDL also lacks empty arrays, variable-sized dynamic arrays (lists), and nested arrays (that is, arrays of arrays are not permitted). The object-oriented features of the language require that the programmer be responsible for managing memory allocation/deallocation. The vectorization is rather primitive, working only along a single axis at a time.

Many historical irregularities survive from the early heritage of the language, requiring individual work-arounds by the programmer. Dynamic typing does not include automatic promotion-on-overflow; one consequence is that (for loops) built in the usual way may fail on the 32,768th iteration. The solution is to define the default type for all integers to 32-bit unsigned (this can be done at startup, or embedded in source code). Array indexing and subroutine entry can both be carried out with exactly the same syntax; this ambiguity, coupled with the single namespace for all variables and subroutines, can cause code to stop working when newly defined subroutines or language extensions conflict with local variable names. IDL programmers can avoid many of these problems by using square brackets for array indexing, thereby avoiding conflicts with function names which use parentheses.

Examples

The following graphics were created with IDL (source code included):

- Image of random data plus trend, with best-fit line and different smoothings
- Plots of delta-o-18 against age and depth (from EPICA and Vostok)

See also

- GNU data language (GDL) - a GNU IDL clone

External link

- Research Systems (http://www.researchsystems.com/)
- NASA GSFC IDL Online Help (http://idlastro.gsfc.nasa.gov/idl_html_help/home.html)
- GDL (GNU data language) web page (http://gnudatalanguage.sourceforge.net/)

Retrieved from "http://en.wikipedia.org/wiki/IDL_programming_language"

Categories: Domain-specific programming languages | Numerical programming languages

- This page was last modified 22:41, 22 September 2005.
- All text is available under the terms of the GNU Free Documentation License (see **Copyrights** for details).
Privacy policy